Snake

Ett klassiskt spel i ny tappning

Helena Harrysson och Linnéa Wickberg

2013-05-20

Handledare Bertil Lindvall

Abstract

The course Digitala Projekt, EITF11, focuses on the process of constructing a prototype for a chosen product. The first step of this process is to specify the requirements of the end product, so that necessary hardware can be chosen, then to connect the different components to each other, so that finally source code can be written. This specific project deals with developing a platform that supports the classical game of Snake. The platform is composed of a processor, an LCD-display, four buttons and two light-emitting diodes. In this report the components are described, as well as the progress of the platform over time. The result shows that the project was successful in developing the platform, and that it supported the version of the game that was written in code.

Innehållsförteckning

Inledning

Kursen Digitala Projekt erbjuder studenter på Industriell Ekonomi möjligheten att i par driva ett projekt som är mer tekniskt intensivt än något av de projekt som tidigare genomförts på utbildningen. Uppgiften består av att studenterna ska ta fram en prototyp av en egenvald produkt, och målet är att ge en större inblick inom konstruktionsarbete och programmering på kärnnivå. Just detta projekt beskriver hur framtagningen av en konsol på vilken man kan spela det klassiska spelet Snake har gått till.

Kravspecifikation

Vårt Snake-spel har följande krav:

- Hårdvaran ska vara korrekt kopplad så att spelet kan styras
- Spelet ska presenteras på en LCD-skärm
- Spelet ska startas genom en startknapp
- Spelaren ska kunna styra ormen med hjälp av två knappar på konsolen
- Höger knapp gör att ormen svänger till höger och vänster knapp får den att svänga åt vänster
- När ormen gått 10 steg får spelaren poäng
- Den aktuella poängställningen ska alltid vara synlig på skärmen
- Då ormen åker in i sig själv eller spelplanens väggar ska spelet avbrytas

Hårdvara

Processor

Till prototypen har en ATMega16 processor använts. Som namnet antyder har den ett minne på 16kB. Den består av 40 pinnar, där 32 av dem är programmerbara I/O pinnar utplacerade på fyra portar (PORTA-PORTD). I vår prototyp har portarna använts på följande sätt:

PORTA: Används för att skicka data mellan processorn och LCD-skärmen. Data till skärmen består av instruktioner gällande var på skärmen något ska ändras, samt om den aktuella databiten ska ha tända eller släckta pixlar. Från skärmen läses en ”busy-flag” för att kontrollera att skärmen är redo att ta emot data från processorn.

PORTB: Används för att skicka instruktioner från processorn till skärmen. Dessa instruktioner påverkar hur skärmen hanterar den data som tas emot från PORTA. Skärmen är till exempel uppdelad i två halvor, och två av pinnarna på denna port avgör vilken av halvorna som ska påverkas av datan.

PORTC: Används för debugging med hjälp av JTAG.

PORTD: Används för att ta in information från knapparna, samt skicka ut information till den röda lysdioden.

Se kopplingsschema i bilaga 1 för att se den kompletta sammankopplingen av delar mellan processorn och övriga komponenter.

LCD-skärm
Spelet i prototypen visas på en grafisk LCD-skärm av typen GDM12864HLCM. Skärmen består av 128x64 pixlar, uppdelat på två halvor om 64x64 pixlar vardera. Koordinatsystemet är lite omvänt det man är van vid då x-koordinaterna går lodrätt och y-koordinaterna vågrätt. X-koordinaterna är dessutom uppdelade register för varje y-koordinat. Ett register avser 8 pixlar, vilket innebär att hantering av skärmen alltid sker i bitar om 8.

Knappar
Spelet manipuleras av spelaren med hjälp av fyra knappar. Två av dem är svarta och används för start av spelet, samt för reset av processorn. De andra två är röda och används för att styra ormen under spelets gång. Knapparna fungerar som så att då de trycks ner skickas spänning vidare till processorn, vilket gör att pinnen går från låg till hög. Då knappen släpps går pinnen tillbaka till låg.

Resistorer
Knapparna är kopplade från processorn till jord, via en resistor. Denna ser till att spänningen till processorn då knappen inte är intryck är 0V, för att försäkra att pinnen är låg vid dessa tillfällen.

JTAG
Med hjälp av vår JTAG kopplades processorn ihop med en dator så att vi i programmet AVR Studio kunde skriva programkod direkt till processorn.

Mjukvara
För att få fram spelet Snake på skärmen har totalt ca 700 rader kod skrivits, men här presenterar vi bara det övergripande tänket och skalet av den. Den kompletta källkoden finns bifogad i bilaga.

Starten av spelet börjar med att initiera skärmen genom att ställa in alla pinnar enligt databladet anvisningar. Nästa metod som körs tömmer först skärmen och ritar sedan ut det som är starläget; nämligen att båda skärmhalvor har en ram och att det på höger halva finns en poängtavla. Efter detta kommer vår eviga while-loop som spelet utgörs av. Den börjar med att lyssna på starknappen, och när den trycks ner kommer ormen fram på skärmen.

Spelet är uppbyggt av många metoder som kallar på varandra, men de som varit viktigast genomgående är commando(), writeData() och delay(). Den första används för att ställa in respektive skärmhalva på det sätt som databladet anger för att skriva till skärmen, vilket sedan görs med metoden writeData(). I dessa metoder kallas även på isBusy() som är en metod som läser av skärmens busy flag. Denna indikerar om skärmen är redo att ta emot ny information från processorn och var därför essentiell för att programmet skulle fungera eftersom processorn arbetar mycket snabbare än skärmen. Detta är också varför metoden delay() används genomgående i koden.

Ormen som ritas ut på skärmen är i koden representerad av en matris, och när ormen går ett steg framåt fylls en plats i matrisen. När tio platser blivit fyllda får spelaren poäng, vilket uppdateras på höger skärmhalva. Själva styrningen är skriven som så att det finns fyra riktningar (1-4) där 1 är rakt uppåt och sen plussas de på motsols. Trycker man på den högra knappen motsvarar det att riktningen ökar med ett, medan vänster innebär att den sjunker med ett.

Om spelaren kör in i en av väggarna eller sig själv så avbryts den eviga loopen och spelomgången är avslutad.

Genomförande

Processen att ta fram en prototyp har bestått av två faser, där den första hade fokus på att konstruera hårdvaran och den andra bestod av skrivande av källkod. För att kunna koppla ihop hårdvarudelarna krävdes först att en kravspecifikation togs fram, i vilken det var beskrivet vad prototypen skulle kunna klara av – vilket avgjorde vilka delar som skulle komma att behövas. Nästa steg var att rita ett kopplingsschema som specificerar precis vilka pinnar på processorn som var kopplade till vilka pinnar på de övriga delarna.

Inte förrän detta var gjort kunde själva bygget börja, och då fick vi våra verktyg och delar. Under cirka två veckors tid kopplades sedan alla delar samman dels via lödning, men också genom att bara vira tråd på pinnarna. Då hårdvaran var färdigkopplad inleddes nästa fas. Inledningsvis gjordes enkla tester för att se att hårdvaran var korrekt kopplad, samt för att förstå hur koden manipulerade portarna på processorn. I samband med detta började vi använda programmet AVR Studio.

Själva kodskrivandet tog sedan fart, små program skrevs för att än en gång se att vi förstod hur förändringar i inställningar visade sig på hårdvaran. För att kunna identifiera förändringarna användes den röda lysdioden genom att vi valde att tända den på olika ställen i koden. Nästa steg var att börja hantera datan och instruktionerna som skickas mellan skärmen och processorn – vilket visade sig vara den svåraste uppgiften hittills. Som inledningen antydde innebar det här projektet en utmaning, och det var när vi skulle försöka förstå hur viktigt det är att alla pinnar är rätt inställda i rätt tid för att kunna läsa och skicka data på rätt sätt som vår tekniska kunskap sattes på prov.

Efter ett antal olika tillämpade metoder började vi till slut känna att det verkade finnas en logisk koppling mellan det vi trodde vi påverkade och vad som faktiskt visade sig på skärmen. Statiska inställningar, såsom utskriven text på höger skärmhalva och en ram runt hela skärmen, såg ut som den skulle och då inleddes arbetet med den dynamiska delen av spelet: själva ormen. Här blev det mycket tänkande för att manipulera rätt pixel då ormen rörde sig, och för att hitta en lösning som inte krävde mer än de 16kB minne vi hade att tillgå.

Till slut hittades en lösning som lät oss spela en version av Snake som uppfyller de krav som ställts i kravspecifikationen. Spelet är inte precis som den klassiska versionen, men vi är väldigt nöjda med att ha tagit fram vår tolkning.

Resultat

Efter vad som känts som en lång process har målet med att ta fram en prototyp lyckats. Efter vissa omformuleringar, helt enligt en sann utvecklingsprocess, har en uppfyller den alla krav satta i kravspecifikationen.

Diskussion

En av anledningarna till att processen känts lång är att den inneburit många motgångar på vägen till följd av okunskap. Inledningsvis fanns förvirring gällande vad en pinne, ett ben, en port och alla möjliga andra bastermer över huvud taget innebar. Allteftersom man förstod det skulle arbetet med att ta fram en kravspecifikation påbörjas och sedan ett kopplingsschema ritas, varpå nya frågor uppstod. Databladen som skulle hjälpa förvirrade nästan ännu mer, mycket eftersom de förutsätter att läsaren har vissa grundkunskaper vilka inte fanns hos oss.

Åtskilliga besök hos vår handledare senare stod kopplingsschemat färdigt och bygget inleddes. I efterhand känns det som att den perioden förflöt smärtfritt, men under tiden kändes det som att man aldrig skulle bli färdig. Då vi upptäckte att något fattades i vårt kopplingsschema, eller att man på annat sätt kunde följa något steg för steg stod vi handfallna och osäkra över vad vi skulle göra. Trots det lyckades vi prestera en hårdvara relativt fort som verkade vara korrekt kopplad.

Inför projektet var vi övertygade om att byggandet av den fysiska produkten skulle vara det som var mest komplicerat eftersom vi aldrig tidigare gjort något liknande, medan programmering i C inte borde vara alltför annorlunda mot den JAVA-programmering vi trots allt läst två kurser i. På sätt och vis stämde den tron; C-programmeringen i sig var inte källan till alltför många problem. Däremot visade det sig vara väldigt komplicerat skriva kod där man hela tiden måste ställa in portar och pinnar i rätt ordning för att något ska hända, samt att ständigt se till så att skärmen är redo att ta emot data från processorn. Dessa delar utgjorde nog den största utmaningen genom hela projektet.

I och med att det känts som vi konstant jobbat i motvind till följd av vår bristande kunskap är vi väldigt stolta över att kunna presentera en fungerande prototyp.

Källförteckning

Datablad för processorn:
http://www.eit.lth.se/fileadmin/eit/courses/edi021/datablad/Processors/ATmega16.pdf

Datablad för LCD-skärmen:
http://www.eit.lth.se/fileadmin/eit/courses/edi021/datablad/Display/GDM12864H.pdf

Kursbok i C-programmering:
Brian W. Kernigan, Dennis M.Ritchie, The C programming language, second edition, Prentice Hall Software Series.

Bilaga 1 – Kopplingsschema